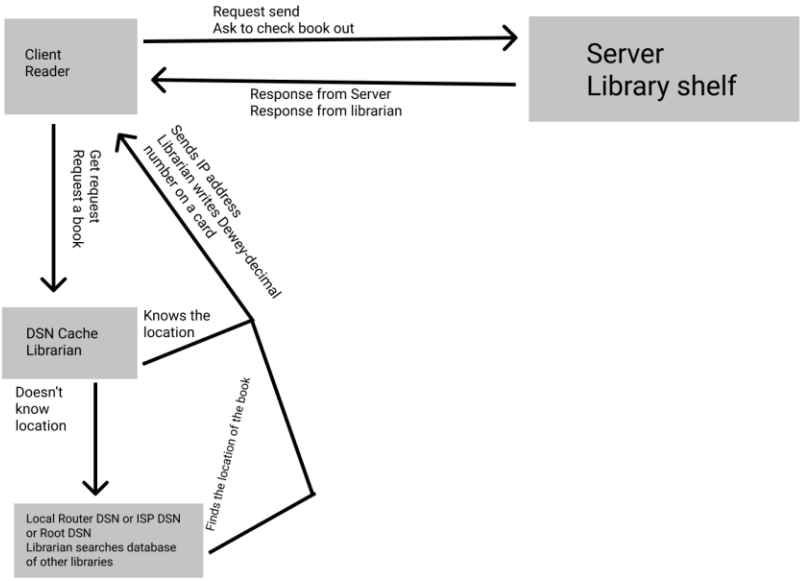
How the Web Works

In this lab, you'll be working with a partner to explore a little more about the internet, the web, requests, responses and more. You'll be reading and writing about concepts as well as practicing some of the commands that we saw during the lecture earlier.

Topic 1: The Internet and the World Wide Web

1) What is the internet? (hint: here)
    a) It is all of the interconnected computers and servers across the world, which use an agreed upon protocols to communicate.
2) What is the world wide web? (hint: here)
    a) Is one of the protocols used within the internet, to send information back and forth via webpages. It is probably the largest or the most well known within the internet, but is separate from email protocols, FTP protocols, and the Telnet.
3) Partner One: read this page on how the internet works, Partner Two: read this page on how the world wide web works. When you're done reading, come back together and and answer the following questions
    a) What are networks?
        i) A bunch of computers hooked up to one another.
    b) What are servers?
        i) A computer that stores files and sends a response when a client computer sends a request.
    c) What are routers?
        i) A small computer that decides which computers in a network gets what information when.
    d) What are packets?
        i) When a small portion of information that is being sent between computers, that can later be pieced together.
4) Come up with a metaphor for the internet and the web, you can do a single one if you think of one that puts them together or two separate ones (feel free to use one you've heard today or read about if you can't think of a new one, but spend at least 10 minutes trying to think of something different before you resort to that)

    It is like a library which stores information, and you have to put in a request to get a certain book. The librarian acts as a router which can figure out where to get the right information. Also, libraries can communicate with other libraries to request information even if it's in another city or state.
5) Draw out a diagram of the infrastructure of the internet and how a request and response travel using your metaphor (like the map and letters we saw during the lecture). Insert the drawing into this document (can be a picture of a physical drawing, a Google Drawing, a Figma drawing, etc)



Topic 2: IP Addresses and Domains

1) What is the difference between an IP address and a domain name?
    a) The IP address, is the specific address of the computer you are requesting information from. The domain name is a more easily remembered name of how to ask a DNS server what the official IP is.
2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal)
    a) 104.22.13.35
3) Try to access devmountain.com by its IP address. It shouldn't work because we have our sites protected by a service called CloudFlare. Why might it be important to not let users access your site directly at the IP address?
    a) If someone can access your website via the IP address, it would likely be more dangerous and easy for them to hack in and get information that should be protection.
4) How do our browsers know the IP address of a website when we type in its domain name? (If you need a refresher, go read this comic linked in the handout from this lecture)
    a) Your browser translates that into an IP address somehow. If you have been to the website before, it might be stored locally via cookies, if not it might be stored on your router, if not your computer will send a request to various DSN servers to ask for the IP address to get it.

Topic 3: How a web page loads into a browser

The steps of how a web page is requested and sent are in the table below. However, **they are out of order**. Unscramble them and explain your thinking/reasoning in the second two columns of the table.

Steps Scrambled	Steps in Correct Order	Why did you put this step in this position?
Example: Here is an example step	*Here is an example step*	*- I put this step first because ____* *- I put this step before/after ____ because ____*
Request reaches app server	Initial request	Nothing happens until the client

		submits a request.
HTML processing finishes	Request reaches app server	The requests goes to the server and initiates the code on server for it to figure out what to do.
App code finishes execution	App code finishes execution	Once it executes the code, it starts sending the response back to the client.
Initial request (link clicked, URL visited)	Browser receives HTML, begins processing	Once the client receives the response, it starts translating the info into a webpage.
Page rendered in browser	HTML processing finishes	It will finish executing the HTML code and after that will display it onto the screen.
Browser receives HTML, begins processing	Page rendered in browser	This is the last step.

Topic 4: Requests and Responses

Setup

- Download the folder for this exercise from Frodo.
- Make sure you unzip it.
- Open it in VS Code
- Run `npm i` in the terminal (make sure you're in the web-works folder you just downloaded).
    - You'll know it was successful if you see a node_modules folder in the web-works folder.
- Run `node server.js` in the terminal (also in the web-works folder) and you should see a log to the terminal saying 'serving up port 4500'
- You'll be using this file to figure out what will happen when you make requests to this server, so read it over to see what's going on. We'll be getting into the two GET functions and the POST function.

Part A: GET /

- You'll start by looking at the function that runs when we make a get request to /, which looks like this: http://localhost:4500 or http://localhost:4500/
- You'll use the curl command to make a request and read the response in your terminal

1) Predict what you'll see as the body of the response:
2) Predict what the content-type of the response will be:

- Open a terminal window and run `curl -i http:localhost:4500`

3) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?
    a) Yes, we predicted it would say the words that were within the HTML code, within the function of the get request, at resource "/" because that was going to be the default landing page for the site.
4) Were you correct about the content-type of the response? If yes, how/why did you make your prediction? If not, what was it and why?
    a) Yes, we predicted that the it would read the response as HTML and the curl command show it as that. With the "<H1> and <H2>" indicators, knowing that that was HTML code helped reach the conclusion.

Part B: GET /entries

- Now look at the next function, the one that runs on get requests to /entries.
- You'll use the curl command again. This time, you'll need to figure out how to modify it to get the response that you need.

1) Predict what you'll see as the body of the response:
    a) We will see the objects within the array variable of "entries."
2) Predict what the content-type of the response will be:
    a) The web browser will just show text on the screen, because it isn't formatted to be HTML.

- In your terminal, run a curl command to get request this server for /entries

3) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?
    a) Yes. It was in the variable at the top of the JS file.
4) Were you correct about the content-type of the response? If yes, how/why did you make your prediction? If not, what was it and why?
    a) No. Instead of just text, it read it as a Json file.

Part C: POST /entry

- Last, read over the function that runs a post request.

1) At a base level, what is this function doing? (There are four parts to this)
    a) It is created a nested variable called "newEntry"
    b) It is pushing that variable to the array of "Entries"
    c) It is adding 1 to the global ID variable
    d) It is sending the array "entries" to the client.
2) To get this function to work, we need to send a body object with our request. Looking at the function in server.js, what properties do you know you'll need to include on that body object? And what data types will they be (hint: look at the objects in the entries array)?
3) Plan the object that you'll send with your request. Remember that it needs to be written as a JSON object inside strings. JSON objects properties/keys and values need to be in **double quotes** and separated by commas.
    a) {"date": "July 19", "content": "I'm so confused."}
4) What URL will you be making this request to?
    a) Localhost? Or 127.0.0.1?
5) Predict what you'll see as the body of the response:
    a) Probably a lot of stuff I won't understand. I have no idea what the client side will see, if anything.
6) Predict what the content-type of the response will be:
    a) Application/json

- In your terminal, enter the curl command to make this request. It should look something like the example below, with the information you decided on in steps 3 and 4 instead of the ALL CAPS WORDS.
    - curl -i -X POST -H 'Content-type: application/json' -d JSONOBJECT URL

7) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why?
    a) I have no idea
    b) Finally figured it out.
8) Were you correct about the content-type of the response? If yes, how/why did you make your prediction? If not, what was it and why?
    a) No clue.
    b) Was confused about the format of the syntax. But essentially it did something similar to what I originally envisioned, it just also gave a bunch of other curl code that was kinda confusing.

Submission

1. Save this document as a PDF
2. Go to Github and create a new repository. (Click the little + in the upper right hand corner.)
3. Name your repository “web-works” (or something like that).
4. Click “uploading an existing file” under the “Quick setup heading”.
5. Choose your web works PDF document to upload.
6. Add “commit message” under the heading “Commit changes”. A good commit message would be something like “Adding web works problems.”
7. Click commit changes.

Further Study: More curl

Visit this link and do the exercises using the website provided. Keep track of the commands you used in this document. (Don’t forget to resubmit to GitHub when you complete this section)